Lecture notes for Jan 23, 2023
Examples of NP-complete problems

Chun-Hung Liu

January 23, 2023

SAT
Input: A formula ϕ.
Output: Determine whether ϕ is satisfiable.

Theorem 1 (Cook) *SAT is* **NP***-complete.*

In fact, a special case of SAT is already **NP**-complete.

k**-SAT**, where k is a fixed positive integer.
Input: A formula ϕ for which every its clause has exactly k literals.
Output: Determine whether ϕ is satisfiable.

Theorem 2 (Karp) *3-SAT is* **NP***-complete.*

We will not prove Theorems 1 and 2 in this course, though the proofs are not very hard. We will only pay attention to use them to prove other problems are **NP**-complete. Usually it is easy to show that a problem is in **NP**. So the difficult part is about how to show the **NP**-hardness.

Lemma 3 *Let D be a decision problem. If there exists an* **NP***-hard problem D' such that there exists a polynomial time reduction from D' to D, then D is* **NP***-hard.*

Proof. Let $A_{D'}$ be a polynomial time reduction from D' to D. Since D' is **NP**-hard, for every problem D'' in **NP**, there exists a polynomial time reduction $A_{D'',D'}$ from D'' to D'. Then the composition of $A_{D'',D'}$ and $A_{D'}$ is a polynomial time reduction from D'' to D. So D is **NP**-hard. ∎

Let's see some examples.

Corollary 4 *For every positive integer $k \geq 3$, k-SAT is* **NP***-complete.*

Proof. It is easy to see that k-SAT is in **NP**. So it suffices to prove that k-SAT is **NP**-hard. We shall find a polynomial reduction from 3-SAT to k-SAT. For every instance I of 3-SAT, we define $f(I)$ to be the formula obtained from I by for each clause of I, duplicating one literal $k-3$ times. Then clearly $f(I)$ is an instance of k-SAT. And it is easy to see that I is satisfiable if and only if $f(I)$ is satisfiable. Clearly $f(I)$ can be constructed in polynomial time in the size of I. So there exists a polynomial time reduction from 3-SAT to k-SAT. Since 3-SAT is **NP**-hard, k-SAT is **NP**-hard by Lemma 3. ∎

A *vertex-cover* of a graph G is a subset S of $V(G)$ such that every edge of G is incident with at least one vertex in S.

VERTEX-COVER
Input: A graph G and an integer k.
Output: Determine whether G has a vertex-cover of size at most k.

Theorem 5 (Karp) *VERTEX-COVER is* **NP***-complete.*

Proof. VERTEX-COVER is in **NP** since for every positive instance (G, k), a vertex-cover of G with size at most k is a certificate that can be verified in polynomial time. So it suffices to prove that it is **NP**-hard.

For every instance I of 3-SAT, say it has m_I clauses and n_I variables, let $k_I = n_I + 2m_I$ and we construction a graph G_I as follows:

- For every variable x in I, create two vertices $v_{x,T}$ and $v_{x,F}$ and add the edge $v_{x,T}v_{x,F}$.
- For every clause c in I and every literal ℓ in c, create a vertex $v_{c,\ell}$; so exactly 3 vertices are created for each clause c; add 3 edges to make those 3 vertices become a clique.

- For each variable x in I and a literal ℓ (say, ℓ is in c), if $\ell = x$, then add the edge $v_{x,T}v_{c,\ell}$; if $\ell = \neg x$, then add the edge $v_{x,F}v_{c,\ell}$.

Clearly G_I and k_I can be constructed in polynomial time in the size of I. So it suffices to prove that I is satisfiable if and only if G_I has a vertex-cover with size at most k_I.

Assume that I is satisfiable. Let g be a truth assignment of I. So for each clause c, there exists a literal ℓ_c that is assigned to be true by g. Let $S_I = \{v_{x,T} : x \text{ is a variable with } g(x) = \text{True}\} \cup \{v_{x,F} : x \text{ is a variable with } g(x) = \text{False}\} \cup \{v_{c,\ell} : c \text{ is a clause, } \ell \text{ is a literal in } c, \ell \neq \ell_c\}$. Then S_I is a subset of $V(G_I)$ with $|S_I| = n_I + (3-1)m_I = k_I$. Clearly, every edge with both ends corresponding to variables or with both ends corresponding to literals is incident with at least one vertex in S_I. For an edge of the form $v_{c,\ell}v_{x,Z}$, where $Z \in \{T, F\}$, if $\ell \neq \ell_c$, then it is incident with a vertex in S_I; otherwise, $\ell = \ell_c$ and $\ell_c \in \{x, \neg x\}$; if $\ell_c = x$, then $v_{x,Z} = v_{x,T}$, and since ℓ_c is true, we know $g(x) = \text{True}$, so $v_{x,Z} = v_{x,T} \in S_I$; if $\ell = \neg x$, then $v_{x,Z} = v_{x,F}$, and since ℓ_c is true, we know $g(x) = \text{False}$, so $v_{x,Z} = v_{x,F} \in S_I$. Hence S_I is a vertex-cover of G_I with size k_I.

Now we assume that G_I has a vertex-cover C with size at most k_I. Let $\mathcal{P}_1 = \{\{v_{x,T}, v_{x,F}\} : x \text{ is a variable in } I\}$ and let $\mathcal{P}_2 = \{\{v_{c,\ell} : \ell \text{ is a literal in } c\} : c \text{ is a clause in } I\}$. Then $\mathcal{P}_1 \cup \mathcal{P}_2$ is a partition of $V(G)$. Since each member M of $\mathcal{P}_1$ is a clique with size 2, C contains at least 1 vertex in M. Since each member M of $\mathcal{P}_2$ is a clique with size 3, C contains at least 2 vertices in M. And $|\mathcal{P}_1| = n_I$ and $|\mathcal{P}_2| = m_I$, so C contains exactly one vertex in each member of $\mathcal{P}_1$ and contains exactly 2 vertices in each member of $\mathcal{P}_2$. For each variable x, if $v_{x,T} \in C$, then we assign x True; if $x_{x,F} \in C$, then assign x False. This is well-defined since C contains exactly 1 vertex in each member of $\mathcal{P}_1$. Since for each clause c, C contains exactly 2 vertices in $\{v_{c,\ell} : \ell \text{ is a literal in } c\}$, there exists an literal ℓ_c^* in c such that $v_{c,\ell_c^*} \notin C$. Let $x_{\ell_c^*}$ be the variable such that $\ell_c^* \in \{x_{\ell_c^*}, \neg x_{\ell,c^*}\}$. So either $\ell_c^* = x_{\ell_c^*}$ and $v_{x,T}v_{c,\ell_c^*} \in E(G_I)$, or $\ell_c^* = \neg x_{\ell_c^*}$ and $v_{x,F}v_{c,\ell_c^*} \in E(G_I)$. Since $v_{c,\ell_c^*} \notin C$ and C is a vertex-cover, we know $v_{x,T} \in C$ for the former case, and $v_{x,F} \in C$ for the latter case. So the literal ℓ_c^* is assigned to be True. Therefore, I is satisfiable.

■

A *stable set* in a graph G is a subset S of $V(G)$ such that vertices in S are pairwise non-adjacent. The *independence number* of G is the maximum size of a stable set in G.

INDEPENDENCE
Input: A graph G and an integer k.
Output: Determine whether G has a stable set of size at least k.

Theorem 6 (Karp) *INDEPENDENCE is* **NP***-complete.*

Proof. INDEPENDENCE is in **NP** since every stable set of size at most k in a positive instance is a certificate that can be verified in polynomial time. To show that INDEPENDENCE is **NP**-hard, by Lemma 3 and Theorem 5, it suffices to show that there exists a polynomial time reduction from VERTEX-COVER to INDEPENDENCE.

Let (G, k) be an instance of VERTEX-COVER. Then $(G, |V(G)| - k)$ is an instance of INDEPENDENCE that can be constructed in polynomial time. Note that a subset S of $V(G)$ is a vertex-cover of G if and only if $V(G) - S$ is a stable set in G. Hence G has a vertex-cover of size at most k if and only if G has a stable set of size at least $|V(G)| - k$. That is, (G, k) is a positive instance of VERTEX-COVER if and only if $(G, |V(G)| - k)$ is a positive instance of INDEPENDENCE. ■

Note that k is part of the input of VERTEX-COVER and INDEPENDENCE. They are no longer hard if k is fixed instead of being part of the input. The reason is that there are at most $|V(G)|^k$ subsets of $V(G)$ of size k, and check each such set is a vertex-cover or a stable set can be done in time $O(k^2)$. So deciding whether a graph has a vertex-cover (or a stable set, respectively) of size k can be solved in time $O(k^2|V(G)|^k)$.

Let G be a graph and let k a positive integer. A *proper k-coloring* of G is a function $f : V(G) \to [k]$ such that for every edge uv of G, $f(u) \neq f(v)$. A graph G is *k-colorable* if there exists a proper k-coloring of G.

k-COLORABILITY, where k is a positive integer
Input: A graph G.
Output: Determine whether G is k-colorable.

Theorem 7 *3-COLORABILITY is* **NP***-complete.*

Proof. Reduced from 3-SAT. The proof is not very difficult, but we omit it here. ■

Theorem 8 *For every positive integer $k \geq 3$, k-COLORABILITY is* **NP***-complete.*

Proof. For every graph G, let G_k be the graph obtained from G by adding $k-3$ new vertices each adjacent to all other vertices of G_k. Then it is easy to show that G is 3-colorable if and only if G_k is k-colorable. Since 3-COLORABILITY is **NP**-hard, k-COLORABILITY is **NP**-hard. And it is easy to show that k-COLORABILITY is in **NP**, so it is **NP**-complete. ■

A graph G is *planar* if it can be drawn in the plane with no edge-crossing.

PLANAR k-COLORABILITY, where k is a positive integer
Input: A graph G.
Output: Determine whether G is k-colorable.

Theorem 9 (Garey, Johnson, Stockmeyer) *PLANAR 3-COLORABILITY is* **NP***-complete.*

Proof. Reduced from 3-COLORABILITY. We omit the proof here. ■

Theorem 9 implies that 3-colorability remains hard even if we strict to the planar graphs. In fact, the proof of Garey, Johnson, Stockmeyer shows that it remains hard even when the planar graphs have maximum degree at most 4. Note that by Brooks' theorem, every planar graph with maximum degree at most 3 is 3-colorable unless K_4 is a component. And K_4 is not 3-colorable. So testing the 3-colorability of a planar graph with maximum degree at most 3 is equivalent to testing whether some of its component is isomorphic to K_4, which can be done in polynomial time obviously.

Note that the famous Four Color Theorem states that every planar graph is 4-colorable. So PLANAR 4-COLORABILITY is in **P**, and k-colorability is no longer algorithmically hard for $k \geq 4$ if we strict to planar graphs. In fact, it is solvable in constant time, because we can always output "yes". Note that it does not contradict to Theorem 8. The graph G_k constructed in the proof of Theorem 8 is not planar.

Finding a proper 4-coloring of a given planar graph is more difficult. But the known proofs of the Four Color Theorem can be transformed into polynomial time algorithms. In particular, the proof of Robertson, Seymour and Thomas gives a quadratic time algorithm.